transportation_analysis

December 19, 2024

1 Transportation Mode Analysis

This notebook analyzes commuting patterns across the United States using the 2023 ACS 5-year estimates dataset.

1.1 Contents

[ ]:
```
# Cell 1: Initial imports
import pandas as pd
import plotly.express as px
import plotly.io as pio
import numpy as np

# Set default renderer for notebook display
pio.renderers.default = 'notebook'

# Cell 2: Read the dataset
df = pd.read_csv('../data/raw/
 ↪NTAD_Means_of_Transportation_to_Work_-4347144958748997132.csv')

# Cell 3: Data Cleaning and Preparation
print("Initial data exploration and cleaning:")
print("-" * 50)
print(f"Original number of records: {len(df)}")
print("\nChecking ALAND values:")
print(df['ALAND'].describe())
print(f"\nNumber of negative ALAND values: {len(df[df['ALAND'] < 0])}")

# Clean the data
df_clean = df.copy()
```

```python
# Remove negative ALAND values
df_clean = df_clean[df_clean['ALAND'] > 0]

# Check for any null values in key columns
transport_cols = [col for col in df_clean.columns if 'Workers' in col]
null_counts = df_clean[transport_cols].isnull().sum()
print("\nNull values in transportation columns:")
print(null_counts[null_counts > 0])

# Remove rows with null values in key columns
df_clean = df_clean.dropna(subset=transport_cols)

# Verify the cleaning results
print("\nAfter cleaning:")
print(f"Number of records: {len(df_clean)}")
print("\nALAND statistics after cleaning:")
print(df_clean['ALAND'].describe())

# Add state names
state_lookup = {
    '01': 'Alabama', '02': 'Alaska', '04': 'Arizona', '05': 'Arkansas',
    '06': 'California', '08': 'Colorado', '09': 'Connecticut',
    '10': 'Delaware', '11': 'District of Columbia', '12': 'Florida',
    '13': 'Georgia', '15': 'Hawaii', '16': 'Idaho', '17': 'Illinois',
    '18': 'Indiana', '19': 'Iowa', '20': 'Kansas', '21': 'Kentucky',
    '22': 'Louisiana', '23': 'Maine', '24': 'Maryland',
    '25': 'Massachusetts', '26': 'Michigan', '27': 'Minnesota',
    '28': 'Mississippi', '29': 'Missouri', '30': 'Montana',
    '31': 'Nebraska', '32': 'Nevada', '33': 'New Hampshire',
    '34': 'New Jersey', '35': 'New Mexico', '36': 'New York',
    '37': 'North Carolina', '38': 'North Dakota', '39': 'Ohio',
    '40': 'Oklahoma', '41': 'Oregon', '42': 'Pennsylvania',
    '44': 'Rhode Island', '45': 'South Carolina', '46': 'South Dakota',
    '47': 'Tennessee', '48': 'Texas', '49': 'Utah', '50': 'Vermont',
    '51': 'Virginia', '53': 'Washington', '54': 'West Virginia',
    '55': 'Wisconsin', '56': 'Wyoming'
}

df_clean['STATEFP'] = df_clean['STATEFP'].astype(str).str.zfill(2)
df_clean['STATE_NAME'] = df_clean['STATEFP'].map(state_lookup)

print("\nVerify state mapping:")
print(df_clean['STATE_NAME'].value_counts().head())
```

```
Initial data exploration and cleaning:
--------------------------------------------------
Original number of records: 85116
```

```
Checking ALAND values:
count    8.511600e+04
mean     4.363422e+07
std      2.588251e+08
min     -2.147484e+09
25%      1.659070e+06
50%      4.389026e+06
75%      2.614074e+07
max      2.146654e+09
Name: ALAND, dtype: float64

Number of negative ALAND values: 625

Null values in transportation columns:
Series([], dtype: int64)

After cleaning:
Number of records: 84393

ALAND statistics after cleaning:
count    8.439300e+04
mean     5.991193e+07
std      1.781612e+08
min      2.128300e+04
25%      1.703139e+06
50%      4.476108e+06
75%      2.678229e+07
max      2.146654e+09
Name: ALAND, dtype: float64

Verify state mapping:
STATE_NAME
California      9073
Texas           6816
New York        5378
Florida         5110
Pennsylvania    3445
Name: count, dtype: int64
```

Show the overall national distribution of transportation modes using our cleaned dataset. Car usage breakdown (drove alone vs carpooled) State-level comparisons Population density analysis

[ ]:
```python
# Calculate national transportation mode totals
total_workers = df_clean['Total workers'].sum()
transportation_modes = {
    'Car (Total)': df_clean['Workers traveling by car, truck, or van'].sum(),
```

```
    'Public Transportation': df_clean['Workers traveling by public␣
 ↪transportation'].sum(),
    'Bicycle': df_clean['Workers traveling by bicycle'].sum(),
    'Walking': df_clean['Workers traveling by walking'].sum(),
    'Other Means': df_clean['Workers traveling by other means'].sum(),
    'Working from Home': df_clean['Workers working from home'].sum()
}

# Create DataFrame for plotting
mode_df = pd.DataFrame({
    'Mode': list(transportation_modes.keys()),
    'Count': list(transportation_modes.values())
})

# Calculate percentages
mode_df['Percentage'] = mode_df['Count'] / total_workers * 100

# Create pie chart
fig = px.pie(
    mode_df,
    values='Count',
    names='Mode',
    title='National Transportation Mode Distribution',
    color_discrete_sequence=px.colors.qualitative.Set1,
    template='plotly_dark'
)

# Update layout with our sleek styling
fig.update_layout(
    title=dict(
        text='National Transportation Mode Distribution',
        x=0.5,
        xanchor='center',
        font=dict(size=24)
    ),
    paper_bgcolor='rgba(0,0,0,0)',
    plot_bgcolor='rgba(0,0,0,0)',
    font=dict(size=14),
    margin=dict(l=50, r=50, t=80, b=50)
)

# Update traces
fig.update_traces(
    textinfo='percent+label',
    hole=0.4,
    marker=dict(line=dict(color='rgb(40,40,40)', width=2))
)
```

```
# Add center annotation
fig.add_annotation(
    text='Transportation<br>Modes',
    x=0.5,
    y=0.5,
    font_size=20,
    showarrow=False
)

# Show the figure
fig.show()

# Save with high resolution
output_path = "../images/national_transportation_distribution_enhanced.png"
pio.write_image(fig, output_path, format="png", scale=2, width=1200, height=800)

# Print the exact percentages
print("\nTransportation Mode Distribution:")
print("-" * 40)
for mode, count in transportation_modes.items():
    percentage = (count / total_workers) * 100
    print(f"{mode}: {percentage:.1f}%")
```

```
Transportation Mode Distribution:
----------------------------------------
Car (Total): 78.8%
Public Transportation: 3.5%
Bicycle: 0.4%
Walking: 2.4%
Other Means: 1.5%
Working from Home: 13.4%
```

[ ]:
```
# Car Usage Analysis: Drove Alone vs Carpooled
car_alone_sum = df_clean['Workers traveling by car, truck, or van - drove␣
 ↪alone'].sum()
car_carpool_sum = df_clean['Workers traveling by car, truck, or van -␣
 ↪carpooled'].sum()

car_data = pd.DataFrame({
   'Category': ['Drove Alone', 'Carpooled'],
   'Count': [car_alone_sum, car_carpool_sum]
})

# Create donut chart
fig = px.pie(
```

```
    car_data,
    values='Count',
    names='Category',
    title='Car Commuting Patterns: Solo vs Carpooling',
    color_discrete_sequence=px.colors.qualitative.Set1,
    template='plotly_dark'
)

# Update layout with our sleek styling
fig.update_layout(
    title=dict(
        text='Car Commuting Patterns: Solo vs Carpooling',
        x=0.5,
        xanchor='center',
        font=dict(size=24)
    ),
    paper_bgcolor='rgba(0,0,0,0)',
    plot_bgcolor='rgba(0,0,0,0)',
    font=dict(size=14),
    margin=dict(l=50, r=50, t=80, b=50)
)

# Update traces
fig.update_traces(
    textinfo='percent+label',
    hole=0.6,
    marker=dict(line=dict(color='rgb(40,40,40)', width=2))
)

# Calculate total car commuters for center annotation
total_car_commuters = car_alone_sum + car_carpool_sum

# Add center annotation
fig.add_annotation(
    text=f'Total Car<br>Commuters<br>{total_car_commuters:,}',
    x=0.5,
    y=0.5,
    font_size=16,
    showarrow=False
)

# Show the figure
fig.show()

# Save with high resolution
output_path = "../images/car_usage_breakdown_enhanced.png"
pio.write_image(fig, output_path, format="png", scale=2, width=1200, height=800)
```

```
# Print detailed statistics
print("\nCar Commuting Statistics:")
print("-" * 40)
print(f"Total Car Commuters: {total_car_commuters:,}")
print(f"Drove Alone: {car_alone_sum:,} ({car_alone_sum/total_car_commuters*100:.
  ↪1f}%)")
print(f"Carpooled: {car_carpool_sum:,} ({car_carpool_sum/
  ↪total_car_commuters*100:.1f}%)")
```

```
Car Commuting Statistics:
----------------------------------------
Total Car Commuters: 124,416,837
Drove Alone: 111,016,528 (89.2%)
Carpooled: 13,400,309 (10.8%)
```

[ ]:
```
# State-level transportation mode analysis
state_modes = df_clean.groupby('STATE_NAME').agg({
    'Workers traveling by car, truck, or van': 'sum',
    'Workers traveling by public transportation': 'sum',
    'Workers traveling by bicycle': 'sum',
    'Workers traveling by walking': 'sum',
    'Workers traveling by other means': 'sum',
    'Workers working from home': 'sum',
    'Total workers': 'sum'
}).reset_index()

# Calculate percentages for each mode
modes = [
    ('car', 'Workers traveling by car, truck, or van'),
    ('transit', 'Workers traveling by public transportation'),
    ('bike', 'Workers traveling by bicycle'),
    ('walk', 'Workers traveling by walking'),
    ('other', 'Workers traveling by other means'),
    ('remote', 'Workers working from home')
]

for short_name, full_name in modes:
    state_modes[f'{short_name}_pct'] = (
        state_modes[full_name] / state_modes['Total workers'] * 100
    ).round(1)

# Create stacked bar chart with improved styling
fig = px.bar(
    state_modes.sort_values('car_pct', ascending=True),  # Sort by car usage
    x=[
```

```
        'car_pct', 'transit_pct', 'bike_pct',
        'walk_pct', 'other_pct', 'remote_pct'
    ],
    y='STATE_NAME',
    orientation='h',
    template='plotly_dark',
    labels={
        'STATE_NAME': 'State',
        'value': 'Percentage of Workers',
        'variable': 'Transportation Mode'
    },
    color_discrete_sequence=px.colors.qualitative.Set1,
    title='Transportation Mode Distribution by State'
)

# Update layout with our custom styling
fig.update_layout(
    title=dict(
        text='Transportation Mode Distribution by State',
        x=0.5,
        xanchor='center',
        font=dict(size=24)
    ),
    paper_bgcolor='rgba(0,0,0,0)',
    plot_bgcolor='rgba(0,0,0,0)',
    font=dict(size=14),
    margin=dict(l=50, r=50, t=80, b=50),
    barmode='stack',
    showlegend=True,
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=-0.2,
        xanchor="center",
        x=0.5,
        title="Transportation Mode",
        font=dict(size=12)
    ),
    height=1200  # Make it taller to accommodate all states
)

# Add grid lines for better readability
fig.update_xaxes(
    gridcolor='rgba(128,128,128,0.2)',
    zeroline=False,
    title_text='Percentage of Workers'
)
```

```
fig.update_yaxes(
    gridcolor='rgba(128,128,128,0.2)',
    zeroline=False,
    title_text='State'
)

# Show the figure
fig.show()

# Save with high resolution
output_path = "../images/state_transportation_distribution_enhanced.png"
pio.write_image(fig, output_path, format="png", scale=2, width=1200,␣
 ↪height=1600)

# Print summary statistics for the top and bottom states by car usage
print("\nStates with Lowest Car Usage:")
print("-" * 40)
low_car = state_modes.nsmallest(5, 'car_pct')
for _, row in low_car.iterrows():
    print(f"{row['STATE_NAME']}:")
    print(f"  Car: {row['car_pct']:.1f}%")
    print(f"  Transit: {row['transit_pct']:.1f}%")
    print(f"  Remote Work: {row['remote_pct']:.1f}%\n")

print("\nStates with Highest Car Usage:")
print("-" * 40)
high_car = state_modes.nlargest(5, 'car_pct')
for _, row in high_car.iterrows():
    print(f"{row['STATE_NAME']}:")
    print(f"  Car: {row['car_pct']:.1f}%")
    print(f"  Transit: {row['transit_pct']:.1f}%")
    print(f"  Remote Work: {row['remote_pct']:.1f}%\n")
```

```
States with Lowest Car Usage:
----------------------------------------
District of Columbia:
  Car: 32.4%
  Transit: 22.3%
  Remote Work: 29.4%

New York:
  Car: 55.8%
  Transit: 22.5%
  Remote Work: 13.4%

Massachusetts:
  Car: 69.6%
```

```
  Transit: 7.0%
  Remote Work: 16.7%

New Jersey:
  Car: 71.4%
  Transit: 8.5%
  Remote Work: 15.0%

Washington:
  Car: 73.3%
  Transit: 4.0%
  Remote Work: 17.7%


States with Highest Car Usage:
----------------------------------------
Mississippi:
  Car: 91.7%
  Transit: 0.3%
  Remote Work: 5.3%

Alabama:
  Car: 89.7%
  Transit: 0.3%
  Remote Work: 7.8%

Arkansas:
  Car: 89.3%
  Transit: 0.3%
  Remote Work: 7.8%

North Dakota:
  Car: 88.2%
  Transit: 0.4%
  Remote Work: 7.0%

Oklahoma:
  Car: 88.1%
  Transit: 0.3%
  Remote Work: 8.5%
```

[ ]:
```
# Calculate population density (workers per square kilometer)
df_clean['population_density'] = df_clean['Total workers'] / (df_clean['ALAND']␣
↪/ 1e6)
```

```
# Remove extreme outliers using IQR method for more reasonable density␣
 ↪categories
Q1 = df_clean['population_density'].quantile(0.25)
Q3 = df_clean['population_density'].quantile(0.75)
IQR = Q3 - Q1
density_cutoff = Q3 + 3 * IQR  # Using 3 times IQR for a more generous cutoff

df_density = df_clean[df_clean['population_density'] <= density_cutoff].copy()

# Create density categories using quantiles
df_density['density_category'] = pd.qcut(
    df_density['population_density'],
    q=4,
    labels=['Low Density', 'Medium-Low Density', 'Medium-High Density', 'High␣
 ↪Density']
)

# Aggregate by density category
density_analysis = df_density.groupby('density_category').agg({
    'Workers traveling by car, truck, or van': 'sum',
    'Workers traveling by public transportation': 'sum',
    'Workers traveling by bicycle': 'sum',
    'Workers traveling by walking': 'sum',
    'Workers traveling by other means': 'sum',
    'Workers working from home': 'sum',
    'Total workers': 'sum'
}).reset_index()

# Calculate percentages
transport_modes = [
    ('Car', 'Workers traveling by car, truck, or van'),
    ('Public Transit', 'Workers traveling by public transportation'),
    ('Bicycle', 'Workers traveling by bicycle'),
    ('Walking', 'Workers traveling by walking'),
    ('Other', 'Workers traveling by other means'),
    ('Remote Work', 'Workers working from home')
]

for mode_name, col_name in transport_modes:
    density_analysis[f'{mode_name} %'] = (
        density_analysis[col_name] / density_analysis['Total workers'] * 100
    ).round(1)

# Create visualization
fig = px.bar(
    density_analysis,
    x='density_category',
```

```
    y=[f'{mode} %' for mode, _ in transport_modes],
    template='plotly_dark',
    title='Transportation Patterns by Population Density',
    labels={
        'density_category': 'Area Density',
        'value': 'Percentage of Workers',
        'variable': 'Transportation Mode'
    },
    color_discrete_sequence=px.colors.qualitative.Set1
)

# Update layout with our styling
fig.update_layout(
    title=dict(
        text='Transportation Patterns by Population Density',
        x=0.5,
        xanchor='center',
        font=dict(size=24)
    ),
    paper_bgcolor='rgba(0,0,0,0)',
    plot_bgcolor='rgba(0,0,0,0)',
    font=dict(size=14),
    margin=dict(l=50, r=50, t=80, b=50),
    barmode='group',  # Use grouped bars for easier comparison
    showlegend=True,
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=-0.2,
        xanchor="center",
        x=0.5,
        title="Transportation Mode"
    )
)

# Add grid lines
fig.update_yaxes(
    gridcolor='rgba(128,128,128,0.2)',
    zeroline=False,
    title_text='Percentage of Workers'
)
fig.update_xaxes(
    gridcolor='rgba(128,128,128,0.2)',
    zeroline=False,
    title_text='Area Density'
)
```

```python
# Show the figure
fig.show()

# Save with high resolution
output_path = "../images/density_transportation_patterns_enhanced.png"
pio.write_image(fig, output_path, format="png", scale=2, width=1200, height=800)

# Print summary statistics
print("\nTransportation Patterns by Population Density:")
print("-" * 50)
for _, row in density_analysis.iterrows():
    print(f"\n{row['density_category']}:")
    for mode, _ in transport_modes:
        print(f"{mode}: {row[f'{mode} %']:.1f}%")
    print(f"Total Workers: {row['Total workers']:,}")
```

```
/var/folders/j7/smpqy2fn30l7j5kcqh76jqp40000gn/T/ipykernel_5897/755852460.py:20:
FutureWarning:

The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.


Transportation Patterns by Population Density:
--------------------------------------------------

Low Density:
Car: 87.1%
Public Transit: 0.4%
Bicycle: 0.2%
Walking: 1.8%
Other: 1.1%
Remote Work: 9.4%
Total Workers: 29,121,666

Medium-Low Density:
Car: 82.7%
Public Transit: 1.0%
Bicycle: 0.2%
Walking: 1.5%
Other: 1.2%
Remote Work: 13.4%
Total Workers: 39,007,997

Medium-High Density:
Car: 80.3%
```

```
Public Transit: 1.8%
Bicycle: 0.3%
Walking: 1.8%
Other: 1.4%
Remote Work: 14.5%
Total Workers: 39,771,835

High Density:
Car: 75.9%
Public Transit: 4.5%
Bicycle: 0.7%
Walking: 2.7%
Other: 1.8%
Remote Work: 14.4%
Total Workers: 41,571,036
```

[ ]:
```python
# First clean the data and create meaningful density categories
df_clean = df.copy()

# Remove negative ALAND values and calculate density
df_clean = df_clean[df_clean['ALAND'] > 0]
df_clean['population_density'] = df_clean['Total workers'] / (df_clean['ALAND']␣
 ↪/ 1e6)  # workers per square km

# Create density categories using quantiles for more meaningful breaks
df_clean['density_category'] = pd.qcut(
    df_clean['population_density'],
    q=4,
    labels=['Low Density (Rural)', 'Medium-Low Density (Suburban)',
            'Medium-High Density (Urban)', 'High Density (City Center)']
)

# Calculate transportation mode percentages by density category
density_analysis = df_clean.groupby('density_category').agg({
    'Workers traveling by car, truck, or van': 'sum',
    'Workers traveling by public transportation': 'sum',
    'Workers traveling by bicycle': 'sum',
    'Workers traveling by walking': 'sum',
    'Workers traveling by other means': 'sum',
    'Workers working from home': 'sum',
    'Total workers': 'sum'
}).reset_index()

# Calculate percentages
transport_modes = [
    ('Car/Truck/Van', 'Workers traveling by car, truck, or van'),
    ('Public Transit', 'Workers traveling by public transportation'),
```

```
    ('Bicycle', 'Workers traveling by bicycle'),
    ('Walking', 'Workers traveling by walking'),
    ('Other Means', 'Workers traveling by other means'),
    ('Remote Work', 'Workers working from home')
]

for short_name, full_name in transport_modes:
    density_analysis[f'{short_name} %'] = (
        density_analysis[full_name] / density_analysis['Total workers'] * 100
    ).round(1)

# Create bar chart
fig = px.bar(
    density_analysis,
    x='density_category',
    y=[f'{mode[0]} %' for mode in transport_modes],
    title='Transportation Patterns by Area Density',
    template='plotly_dark',
    labels={
        'density_category': '',
        'value': 'Percentage of Workers',
        'variable': 'Transportation Mode'
    },
    color_discrete_sequence=px.colors.qualitative.Set1
)

# Update layout
fig.update_layout(
    title=dict(
        text='How Transportation Choices Change with Population Density',
        x=0.5,
        xanchor='center',
        font=dict(size=24)
    ),
    paper_bgcolor='rgba(0,0,0,0)',
    plot_bgcolor='rgba(0,0,0,0)',
    font=dict(size=14),
    margin=dict(l=50, r=50, t=80, b=50),
    barmode='group',  # Use grouped bars for easier comparison
    showlegend=True,
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=-0.2,
        xanchor="center",
        x=0.5,
        title="Transportation Mode"
```

```
    )
)

# Add grid lines
fig.update_yaxes(
    gridcolor='rgba(128,128,128,0.2)',
    zeroline=False,
    title_text='Percentage of Workers'
)

# Show the figure
fig.show()

# Save
output_path = "../images/density_transportation_patterns_enhanced.png"
pio.write_image(fig, output_path, format="png", scale=2, width=1200, height=800)
```

```
/var/folders/j7/smpqy2fn30l7j5kcqh76jqp40000gn/T/ipykernel_5897/2512922628.py:17
: FutureWarning:

The default of observed=False is deprecated and will be changed to True in a
future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
```

[ ]: